**Program4**

from sklearn.datasets import load\_iris

from sklearn.model\_selection import train\_test\_split, GridSearchCV

from sklearn.tree import DecisionTreeClassifier, plot\_tree

from sklearn.metrics import accuracy\_score, classification\_report

import matplotlib.pyplot as plt

# Load the Iris dataset

data = load\_iris()

X = data.data

y = data.target

# Split the data into training and testing sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3, random\_state=42)

# Step 1: Train a basic Decision Tree model

default\_tree = DecisionTreeClassifier(random\_state=42)

default\_tree.fit(X\_train, y\_train)

plt.figure(figsize=(12, 8))

plot\_tree(default\_tree, feature\_names=data.feature\_names, class\_names=data.target\_names, filled=True)

plt.title("Basic Decision Tree")

plt.show()

# Make predictions and evaluate the model

default\_predictions = default\_tree.predict(X\_test)

default\_accuracy = accuracy\_score(y\_test, default\_predictions)

print("Default Decision Tree Accuracy:", default\_accuracy)

print("\nClassification Report for Basic Model:\n")

print(classification\_report(y\_test, default\_predictions))

# Step 2: Hyperparameter tuning using GridSearchCV

param\_grid = {

    'criterion': ['gini', 'entropy'],

    'max\_depth': [None, 2, 4, 6, 8, 10],

    'min\_samples\_split': [2, 5, 10],

    'min\_samples\_leaf': [1, 2, 4]

}

grid\_search = GridSearchCV(DecisionTreeClassifier(random\_state=42), param\_grid, cv=5, scoring='accuracy')

grid\_search.fit(X\_train, y\_train)

# Best parameters and model

best\_params = grid\_search.best\_params\_

best\_tree = grid\_search.best\_estimator\_

# Make predictions with the tuned model

tuned\_predictions = best\_tree.predict(X\_test)

tuned\_accuracy = accuracy\_score(y\_test, tuned\_predictions)

print("\nBest Parameters:", best\_params)

print("Tuned Decision Tree Accuracy:", tuned\_accuracy)

# Step 3: Visualize the Decision Tree

plt.figure(figsize=(12, 8))

plot\_tree(best\_tree, feature\_names=data.feature\_names, class\_names=data.target\_names, filled=True)

plt.title("Tuned Decision Tree")

plt.show()

# Step 4: Detailed evaluation

print("\nClassification Report for Tuned Model:\n")

print(classification\_report(y\_test, tuned\_predictions))